A bio-plausible design for visual pose stabilization

Shuo Han, Andrea Censi, Andrew D. Straw, and Richard M. Murray

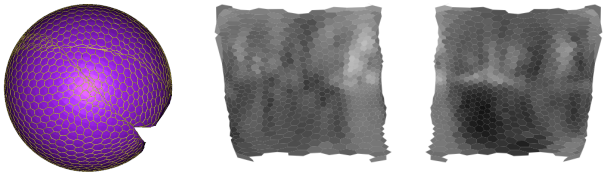
***Abstract*—We consider the problem of purely visual pose stabilization (also known as *servoing*) of a second-order rigid-body system with six degrees of freedom: how to choose forces and torques, based on the current view and a memorized goal image, to steer the pose towards a desired one. Emphasis has been given to the bio-plausibility of the computation, in the sense that the control laws could be in principle implemented on the neural substrate of simple insects. We show that stabilizing laws can be realized by bilinear/quadratic operations on the visual input. This particular computational structure has several numerically favorable characteristics (sparse, local, and parallel), and thus permits an efficient engineering implementation. We show results of the control law tested on an indoor helicopter platform.**

I. Introduction

The question of how the brain utilizes millions of neurons, presumably in a highly parallel fashion, to make timely decisions from information-rich sensory inputs still remains unanswered. This holds even for one of the simplest animals, the fruit fly, which has only about 300,000 neurons, orders of magnitude fewer than the human brain. Engineers marvel at the series of *fast* and *robust* behaviors implemented on the poor computational support of *slow* and *noisy* neurons. Particularly interesting is the processing of visual information (Fig. 1), which biologists believe is a dominant sense controlling flight in many species of insects. Examples of classical studied behaviors include turning towards features of interest [1] and altitude/velocity control [2]. In this paper, we are especially interested in the visual behaviors that allow insects to return to or to stay at a particular location in space (*hovering* and *homing*).

The hovering and homing behaviors are instances of what in robotics is called *visual servoing*, which we will refer to as *pose stabilization*. Our goal in this paper is to solve this problem while respecting the constraints of a "bio-plausible" computation. The motivation is twofold: on the one hand, this might provide clues for realizing a computational model of the insects behavior, useful to biologists; on the other hand, it might inspire useful engineering principles to realize similar robust systems. The question of what can be considered bio-plausible, is, of course, loosely defined. In brief, we consider something to be bio-plausible if it can be realized on neural circuits [3]. The computation should be implementable using parallel asynchronous units; in insects, a model of computation that is well accepted as bio-plausible is a local nonlinear operation followed by wide-field linear integration. An example of computation that could not be considered bio-plausible would be (point) feature-based visual servoing: not as much for implementing a feature-extraction algorithm, but because the logic of manipulating a "list of features" and matching them does not appear to be easily implementable on a neural substrate. An alternative is using the raw luminance, which, interestingly, is a new trend in robotics [4]–[7]. The feasibility of this scheme for insects has been considered, and an experimental assessment concluded that "view-based homing with panoramic images is in principle feasible in natural environments and does not require the identification of individual landmarks" [8].

The authors are with the Division of Engineering and Applied Science, California Institute of Technology, Pasadena, CA 91125, USA. {hanshuo, astraw}@caltech.edu, {andrea, murray}@cds.caltech.edu



(a) (b)

Figure 1. (a) The spatial disposition of ommatidia ("pixels") and (b) an example of simulated visual input (rendered using the Mercator projection) for a fruit fly. The total 1,398 ommatidia of the two compound eyes cover almost the entire visual sphere. Despite being noisy and low-resolution, vision is known as a dominant sense for flight stabilization in fruit flies.

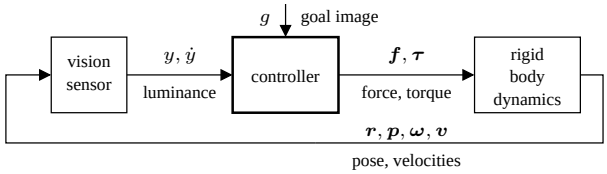


Figure 2. Purely visual pose stabilization. We control in force and torque to stabilize a rigid body based only on the visual input y and $\dot{y}$, and a memorized goal image g taken at the desired pose. No inertial or velocity sensors are used.

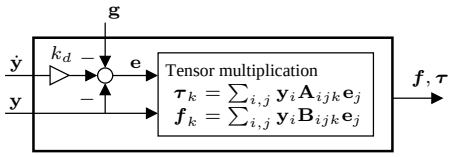


Figure 3. Proportional-derivative (PD) controller using (discrete) tensor multiplications between the visual input y, $\dot{y}$, and the goal image g. This realization is equivalent to a wide-field integration of local nonlinearities, which is an accepted bio-plausible model.

Related work: Visual servoing is a classical problem in industrial robotics, with applications in controlling robot manipulators [9], [10] or visual navigation [11], [12]. The classical approach normally consists of extracting and tracking fiducial point/segments from the image [13]. However, in the last few years, several methods have been proposed that do not rely on features, but instead use the raw luminance directly [4]–[7]. These are formulated as computing the gradient of an image dissimilarity measure; the obvious but not unique choice is the squared error over the field of view. An issue that appears not

to be completely solved at the moment is formulating sufficient conditions for convergence, especially because the gradient directions depend on the distance to the objects, which is usually unknown and thus must be approximated.

Contribution: In this paper, we present proportional-derivative control laws that solve the visual servoing problem for a second-order system controlled in force/torque and demonstrate the pose stabilization results on an indoor helicopter testbed. We do not rely on velocity or inertial sensors; we only assume to know the current visual observations and a goal image taken at the desired pose (Fig. 2). The computation essentially consists of a tensor multiplication that involves the goal image and the current observations (Fig. 3). This paper is the ideal continuation of [14], where we restricted ourselves to purely rotational motion. The main improvement in this paper is that we now solve the problem for joint rotation and translation instead of pure rotation. This is not a trivial extension, because when performing translation, the visual observation will be affected by the distance to objects, which is not instantaneously observable.

With respect to related work in engineering on feature-free visual servoing, we offer the following contributions:

- We consider a second-order system: the control inputs are torque/force rather than velocities; this requires estimating velocity from vision to create suitable damping laws.
- We prove the stability of purely visual control laws independent of (or robust to changes in) the distance profile, given that the environment satisfies certain conditions; previous work did propose laws assuming constant distance, but did not prove convergence (e.g., [6]) or proved it assuming the availability of point features (e.g., [15]).

The paper is organized as follows. Section II introduces the notation used and preliminaries about the model. Section III–V show, in the order of complexity: 1) control laws for the first-order system, controlled in velocity, 2) damping control laws that stabilize the velocities to zero, and 3) proportional-derivative control of the second-order system. Section VI discusses the computational structure of the proportional-derivative control law and several of its favorable numerical features that allow an efficient implementation. Section VII shows numerical tests using simulated visual input of a fruit fly and discusses the bio-plausibility of the control law. Section VIII presents experiments on an indoor helicopter testbed.

II. Preliminaries

We consider the general 6-degree-of-freedom (6-DOF) pose stabilization problem using the output from a generalized vision sensor. In the section, we will introduce the model of the system dynamics, describe the model of the vision sensor, and formalize the visual control problem.

A. Dynamic model

We consider the fully actuated rigid body motion on the special Euclidean group $\mathrm{SE}(3)$. A pose in $\mathrm{SE}(3)$ can be described by the body position $\boldsymbol{p} \in \mathbb{R}^3$ and the body attitude $\boldsymbol{r}$, represented as a 3×3 rotation matrix, or, more formally, an element in the special orthogonal group $\mathrm{SO}(3)$.

For many traditional visual servoing applications where the velocities can be controlled directly, it suffices to consider the system kinematics (also referred to as the first-order system):

$$\begin{cases} \dot{\boldsymbol{r}} = \boldsymbol{r}\,(\boldsymbol{\omega})^\wedge, \\ \dot{\boldsymbol{p}} = \boldsymbol{r}\boldsymbol{v}, \end{cases} \tag{1}$$

where $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity and $\boldsymbol{v} \in \mathbb{R}^3$ is the linear velocity, both expressed in the body frame. The "hat map" $(\cdot)^\wedge$ maps a vector $\boldsymbol{x} \in \mathbb{R}^3$ to a skew-symmetric matrix $\boldsymbol{x}^\wedge \in \mathbb{R}^{3\times 3}$ such that $(\boldsymbol{x}^\wedge)\boldsymbol{y} = \boldsymbol{x} \times \boldsymbol{y}$, with "$\times$" being the ordinary cross product in $\mathbb{R}^3$ [16]. Control in velocity will be discussed primarily in Section III only.

In other applications, such as stabilizing aerial vehicles, where the control inputs are torques and forces, the full dynamics (also referred to as the second-order system) need to be considered:

$$\begin{cases} \dot{\boldsymbol{r}} &= \boldsymbol{r}\,(\boldsymbol{\omega})^\wedge, \\ \mathbb{I}\dot{\boldsymbol{\omega}} &= (\mathbb{I}\boldsymbol{\omega}) \times \boldsymbol{\omega} + \boldsymbol{\tau}, \\ \dot{\boldsymbol{p}} &= \boldsymbol{r}\boldsymbol{v}, \\ m\dot{\boldsymbol{v}} &= m\boldsymbol{v} \times \boldsymbol{\omega} + \boldsymbol{f}, \end{cases} \tag{2}$$

where $\mathbb{I} \in \mathbb{R}^{3\times 3}$ is the moment of inertia; $m \in \mathbb{R}$ is the body mass; $\boldsymbol{\tau} \in \mathbb{R}^3$ is the control torque; $\boldsymbol{f} \in \mathbb{R}^3$ is the control force. Most of this paper will address the issue of controlling such systems. Although this model does not fully capture the dynamics of some platforms such as a helicopter (underactuated, as will be discussed in Section VIII), it provides a good approximation at least near the equilibrium (i.e., at hovering).

B. Sensor model

We assume to have available only the output of a vision sensor. With good generality, we model the vision sensor as a device that at each time t returns a series of values $y(s_i, t)$, each corresponding to the observed luminance from a pixel i in the direction $s_i \in \mathbb{S}^2$. Here $\mathbb{S}^2 \subset \mathbb{R}^3$ denotes the unit sphere. There are two important aspects of such a model:

- *Sensor output*: The model gives *raw luminance* as the output. As a comparison, traditional visual servoing literature often assumes to know the positions of fiducial points on the image plane (except in, e.g., [4]–[7]). The formulation also works if an instantaneous point-wise filter, such as contrast normalization, is applied to the raw luminance.
- *Sensor geometry*: Using the directions $s_i \in \mathbb{S}^2$ to model the disposition of "pixels" on the sensor makes the model equally apt for a normal perspective camera, a catadioptric camera, or the compound eye of a fruit fly (see Fig. 1). This modeling scheme is becoming popular in recent visual servoing as well as other visual control literature, where omnidirectional cameras are preferred since they mitigate the issue of partial observation.

In deriving the theory, we make a number of simplifying assumptions: 1) We are able to sample the entire visual field from "a continuum of pixels", i.e., we know $y(s,t)$ for all $s \in \mathbb{S}^2$; 2) We can observe both $y(t)$ and $\dot{y}(t)$; 3) We ignore the effect of occlusions. Note that, however, in the simulations we do incorporate non-ideal factors such as blurring, sensor dynamics, and occlusions, some of which appear in the experiments as well. We will also discuss how to approximate/discretize the control law when the visual field is discrete later in Section VI.

Given a certain environment, the visual observation $y(s,t)$ will change with the pose of the agent. In general, however, it is not possible to give a closed-form expression of $y(s,t)$ as a function of the pose, because it depends, in a complicated way, on the "nearness" $\mu(s,t)$, defined as the inverse of the distance to the object in direction s. Fortunately, we will only need the relation between $\dot{y}(s,t)$ and $\mu(s,t)$, which is given by the optic flow equation:

$$\dot{y}(s,t) = \mu(s)\nabla_s y(s,t)^* \boldsymbol{v} + (s \times \nabla_s y(s,t))^* \boldsymbol{\omega}, \qquad (3)$$

or more compactly, dropping s and t, and defining the operator S as: $\mathrm{S}y \triangleq s \times \nabla_s y$,

$$\dot{y} = \mu \left(\nabla y\right)^* \boldsymbol{v} + \left(\mathrm{S}y\right)^* \boldsymbol{\omega}. \qquad (4)$$

The two terms in $\dot{y}$ describes the contributions to change in luminance by translational velocity $\boldsymbol{v}$ and rotational velocity $\boldsymbol{\omega}$, respectively. This is not a new result and has appeared, often in a disguised form with different notation, in many other papers. Here we have adopted the convention that ∇y, when evaluated at a specific $s \in \mathbb{S}^2$, is an element of $\mathbb{R}^3$; i.e., we think of it as an "arrow attached to the sphere $\mathbb{S}^2$".

C. Control problem

We assume that the agent knows a "goal" image $g(s)$ taken at the goal pose $\boldsymbol{q}_g = (\boldsymbol{r}_g, \boldsymbol{p}_g)$. The problem we wish to solve can be stated as follows:

Problem 1 (Visual pose stabilization): Given the goal image g, design a stabilizing control law for $\boldsymbol{\tau}$ and $\boldsymbol{f}$, depending only on y, $\dot{y}$, g, such that $\boldsymbol{q} \to \boldsymbol{q}_g$ and $(\boldsymbol{\omega}, \boldsymbol{v}) \to 0$.
Without loss of generality, we will assume that $(\boldsymbol{r}_g, \boldsymbol{p}_g) = (\mathrm{Id}, 0)$, the identity element of $\mathrm{SE}(3)$.

III. Visual control in velocity

We first consider the problem of controlling the first-order system (1), assuming that we can impose $\boldsymbol{\omega}$ and $\boldsymbol{v}$ directly. Motivated by a control law constructed from gradient descent, we obtain a stabilizing control law that does not rely on knowledge of the structure of the environment (i.e., the nearness $\mu(s)$). Not only will the result be useful for controlling the first-order system (1) *per se*, it will also be important in constructing a stabilizing control law for the full second-order system (2), as will be discussed in Section V.

A. Control with the knowledge of distance

A natural way to define a stabilizing control law is to use the *gradient* of the quadratic cost function J defined in the space of images:

$$J(\boldsymbol{q}) = \frac{1}{2}\int_{s\in\mathbb{S}^2} (y(s) - g(s))^2 d\mathcal{S}. \qquad (5)$$

This integration is taken over the entire visual sphere $\mathbb{S}^2$, with $\mathcal{S}$ as the unique rotation-invariant measure on $\mathbb{S}^2$. We will use a shorthand notation to denote this integration:

$$\langle f \rangle \triangleq \int_{s\in\mathbb{S}^2} f(s)\, d\mathcal{S}.$$

Using this notation, equation (5) can be written compactly as

$$J(\boldsymbol{q}) = \tfrac{1}{2}\left\langle (y-g)^2 \right\rangle. \qquad (6)$$

Here, the cost function J is expressed as a function of $\boldsymbol{q}$ because y varies with $\boldsymbol{q}$. Again, note that J is computed directly from the luminance, not from the error in the positions of point features.

However, simply applying the (negative) gradient as the control input may fail to drive the pose $\boldsymbol{q}$ to $\boldsymbol{q}_g$: as an extreme case, an environment with uniform luminance ($y(s) = \mathrm{const.}$) will make $\boldsymbol{q}$ unobservable from the visual input y alone. Therefore, some additional condition on the environment is expected for the gradient control law to work, which is stated formally in the following proposition.

Proposition 1 (Exact gradient): The gradient control law

$$\begin{cases} \boldsymbol{\omega} = & \langle (\mathrm{S}y)(g-y) \rangle, \\ \boldsymbol{v} = & \langle \mu(\nabla y)(g-y) \rangle \end{cases} \qquad (7)$$

locally stabilizes the first-order system (1) asymptotically if the 6×6 "contrast matrix"

$$C(\boldsymbol{q}) = \begin{bmatrix} \langle \mathrm{S}y\mathrm{S}y^* \rangle & \langle \mu \mathrm{S}y \nabla y^* \rangle \\ \langle \mu \nabla y \mathrm{S}y^* \rangle & \langle \mu^2 \nabla y \nabla y^* \rangle \end{bmatrix}$$

is strictly positive definite at $\boldsymbol{q} = \boldsymbol{q}_g$. Note again we have used the convention that both $\mathrm{S}y$ and ∇y (evaluated at a specific s) are vectors in $\mathbb{R}^3$.

Proof: We first note that we have to deal with quantities living on manifolds. Therefore there is some additional difficulty with respect to Euclidean spaces. For example, the gradient is not "a vector in $\mathbb{R}^n$", and the Hessian is not "a matrix in $\mathbb{R}^{n\times n}$". Our first step is a change of coordinates for the attitude. There are many choices of Euclidean coordinates for $\mathrm{SO}(3)$, including Euler angles, quaternions, etc. Here we choose the exponential coordinates $\boldsymbol{\varphi} \in \mathbb{R}^3$ due to its compact written form:

$$\boldsymbol{\varphi} = \mathrm{Log}(\boldsymbol{r})^\vee, \qquad \boldsymbol{r} = \mathrm{Exp}(\boldsymbol{\varphi}^\wedge),$$

where Log and Exp are the *matrix* logarithm and exponential, and $(\cdot)^\vee$ is the inverse of the hat map $(\cdot)^\wedge$. Note in particular that $(\boldsymbol{r} = \mathrm{Id}) \Leftrightarrow (\boldsymbol{\varphi} = 0)$. Under such a representation, the kinematics (1) can be rewritten as

$$\begin{cases}\dot{\boldsymbol{\varphi}} = \boldsymbol{\omega}, \\ \dot{\boldsymbol{p}} = \mathrm{Exp}(\boldsymbol{\varphi}^\wedge)\boldsymbol{v}. \end{cases} \tag{8}$$

Note that both $\boldsymbol{\omega}$ and $\boldsymbol{v}$ depend on y, which is a function of $\boldsymbol{\varphi}$ and $\boldsymbol{p}$. To prove local stability, it suffices to investigate the linearization (see Appendix A) of the system (8) around the equilibrium $(\boldsymbol{\varphi}, \boldsymbol{p}) = (0, 0)$:

$$\dot{\boldsymbol{\xi}} = -C(\boldsymbol{q}_g)\boldsymbol{\xi}, \qquad \boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\varphi} \\ \boldsymbol{p} \end{bmatrix}, \tag{9}$$

where C is the "contrast matrix". Therefore, the system is locally asymptotically stable if $C(\boldsymbol{q}_g) > 0$. ∎

The above condition on the contrast matrix imposes requirements on the visual appearance (reflected in the visual input y) and structure (reflected in the nearness profile μ) of the environment. If the condition fails to hold (i.e., the matrix C loses rank), there exists a certain "direction" in $\mathrm{SE}(3)$ such that the visual observation will not change when moving along this direction infinitesimally near the goal $\boldsymbol{q}_g$. In other words, the environment possesses some kind of spatial invariance that makes Problem 1 impossible to solve with visual input.

Alternatively, we note that the contrast matrix C is in fact the Hessian of J, expressed in the exponential coordinates $\boldsymbol{\varphi}$ and $\boldsymbol{p}$. The Hessian of J being strictly positive definite at $\boldsymbol{q}_g$ ensures that $\boldsymbol{q}_g$ is an isolated minimum of $J(\boldsymbol{q})$. This means that $\boldsymbol{q} = \boldsymbol{q}_g$ is the only (local) solution of the equation $y(s) = g(s)$, which is equivalent to $\boldsymbol{q}_g$ being observable from the visual input y.

B. Control without the knowledge of distance

One drawback of the above control law (7) is that $\boldsymbol{v}$ depends on the unknown nearness profile $\mu(s)$. A possible solution would be to estimate μ by solving a *structure from motion* problem (e.g., [17]–[19]). Our approach, instead, will show that it is possible to use a control law that does not depend on μ, if we are only concerned with local stability near $\boldsymbol{q}_g$. Since this control law does not contain μ, it is only an approximation of the exact gradient direction, and requires a different condition other than that in Proposition 1 to hold in order to guarantee local stability:

Proposition 2 (Approximate gradient): The approximate gradient control law

$$\begin{cases}\boldsymbol{\omega} = \langle (\mathrm{S}y)(g-y) \rangle, \\ \boldsymbol{v} = \alpha \langle (\nabla y)(g-y) \rangle \end{cases} \tag{10}$$

locally stabilizes the first-order system asymptotically, if there exists $\alpha > 0$ such that the 6×6 "modified contrast matrix"

$$\widetilde{C}(\boldsymbol{q}, \alpha) \triangleq \begin{bmatrix} \langle \mathrm{S}y\mathrm{S}y^* \rangle & \left\langle \left(\frac{\alpha}{2} + \frac{\mu}{2}\right) \mathrm{S}y \nabla y^* \right\rangle \\ \left\langle \left(\frac{\alpha}{2} + \frac{\mu}{2}\right) \nabla y \mathrm{S}y^* \right\rangle & \alpha \langle \mu \nabla y \nabla y^* \rangle \end{bmatrix}$$

is strictly positive definite at $\boldsymbol{q} = \boldsymbol{q}_g$. Note that α is a constant scalar and does not depend on the directions s as opposed to $\mu(s)$ in (7).

Proof: The proof of stability is similar to the previous one. The linearization (see Appendix A) of the system (8) is: $\dot{\boldsymbol{\xi}} = -F\boldsymbol{\xi}$, where

$$F = \begin{bmatrix} \langle \mathrm{S}g\mathrm{S}g^* \rangle & \alpha \langle \nabla g \mathrm{S}g^* \rangle \\ \langle \mu \nabla g \mathrm{S}g^* \rangle & \alpha \langle \mu \nabla g \nabla g^* \rangle \end{bmatrix}. \tag{11}$$

Note that, at $\boldsymbol{q} = \boldsymbol{q}_g$, the modified contrast matrix $\widetilde{C}$ is the symmetric part of F: $\widetilde{C}(\boldsymbol{q}_g, \alpha) = \frac{1}{2}(F + F^*)$. If $\widetilde{C} > 0$, stability can be immediately concluded from the Lyapunov function $V = \|\boldsymbol{\xi}\|^2$. ∎

Remark 1: Under at least some special cases, the positive definiteness of $\widetilde{C}(\boldsymbol{q}, \alpha)$ is equivalent to the positive definiteness of $C(\boldsymbol{q})$. For example, when the environment is spherical ($\mu(s) = \overline{\mu} > 0$), if we let $\alpha = \overline{\mu}$ in $\widetilde{C}(\boldsymbol{q}, \alpha)$, then $\widetilde{C} = C$. This is expected because knowledge of the nearness μ becomes unimportant for a spherical environment.

IV. Visual velocity damping

Our ultimate goal is to construct a proportional-derivative controller, which is necessary to achieve asymptotic stability of the second-order system (2). The result from the previous section is derived from the gradient of the quadratic error metric, and thus can be viewed as the proportional part (i.e. "proportional" to the error in pose). In this section, we switch the focus to designing the derivative part (i.e. the time derivative of the pose), in order to introduce *artificial damping* to the rigid body dynamics. Equivalently, this means one needs to choose force and torque such that the velocities $\boldsymbol{\omega}$, $\boldsymbol{v}$ are driven to 0. Traditionally, the derivative part is constructed to be proportional to the velocities estimated from sensory input, here being y and $\dot{y}$. We will first examine the least-squares estimates, but will soon switch to a much simpler solution using bilinear forms. Despite the simplicity of the latter solution, the resulting control law is effective in that it will stabilize the velocities to 0.

A. Velocity estimation using least squares

The sensory input that encodes information on the velocities is the temporal change of luminance $\dot{y}$. Recall the relationship between $\dot{y}$ and $\boldsymbol{\omega}$, $\boldsymbol{v}$ in (4) and note that $\dot{y}$ is linear in $\boldsymbol{\omega}$, $\boldsymbol{v}$, respectively. In the pure rotation ($\boldsymbol{v} = 0$) or pure translation ($\boldsymbol{\omega} = 0$) case, a straightforward solution is to estimate the velocities directly using least squares:

Lemma 1: The least-squares estimates of the velocities are

$$\hat{\boldsymbol{\omega}}_{\mathrm{LS}} = \langle (\mathrm{S}y)(\mathrm{S}y)^* \rangle^{-1} \langle (\mathrm{S}y)\dot{y} \rangle \quad (\boldsymbol{v} = 0),$$
$$\hat{\boldsymbol{v}}_{\mathrm{LS}} = \langle (\mu \nabla y)(\mu \nabla y)^* \rangle^{-1} \langle (\mu \nabla y)\dot{y} \rangle \quad (\boldsymbol{\omega} = 0).$$

Although the above only holds for separate motions and will not be our final solution, they provide intuition for the bilinear velocity estimation introduced in the next.

B. Velocity estimation using bilinear forms

For control purposes, it is likely that we do not need to estimate the velocities perfectly, since the feedback loop can usually tolerate some uncertainties. We now attempt to simplify the above velocity estimates, hoping to make the computation more bio-plausible. First, we will remove the matrix inverse from the least-squares estimates. Second, motivated by the previous section, we will drop the (unknown) nearness $\mu(s)$. This gives the following velocity estimates that are bilinear (denoted as "BL") forms of y and $\dot{y}$, namely, linear in y and $\dot{y}$ separately:

$$\hat{\boldsymbol{\omega}}_{\text{BL}} \triangleq \langle (\text{S}y)\, \dot{y} \rangle\,, \qquad \hat{\boldsymbol{v}}_{\text{BL}} \triangleq \langle (\nabla y)\, \dot{y} \rangle\,. \tag{12}$$

Surprisingly, as we will see, despite such significant simplification, they do not prevent us from obtaining a stabilizing damping control law. One insight is that the resulting approximate velocity estimates are "good enough" because they will remain "close enough" (or more precisely, within 90 deg) from the true velocities (in the case of separate motions):

Proposition 3: In the case of separate motions, the bilinear velocity estimates are related to the true velocities as

$$\boldsymbol{\omega}^* \hat{\boldsymbol{\omega}}_{\text{BL}} \geq 0, \qquad \boldsymbol{v}^* \hat{\boldsymbol{v}}_{\text{BL}} \geq 0.$$

Proof: For pure rotation, $\dot{y} = (\text{S}y)^*\, \boldsymbol{\omega}$, and $\boldsymbol{\omega}^* \hat{\boldsymbol{\omega}}_{\text{BL}} = \boldsymbol{\omega}^* \langle (\text{S}y)\, \dot{y} \rangle = \boldsymbol{\omega}^* \left\langle (\text{S}y)\, (\text{S}y)^* \right\rangle \boldsymbol{\omega} \geq 0$.

Similarly, for pure translation, $\dot{y} = \mu\, (\nabla y)^*\, \boldsymbol{v}$, and $\boldsymbol{v}^* \hat{\boldsymbol{v}}_{\text{BL}} = \boldsymbol{v}^* \langle (\nabla y)\, \dot{y} \rangle = \boldsymbol{v}^* \langle \mu \nabla y \nabla y^* \rangle\, \boldsymbol{v} \geq 0$. ∎

Omitting the matrix inverse makes the velocity estimator *arbitrarily inaccurate* in scale, without more constraints on the environment. In fact, suppose that $\hat{\boldsymbol{\omega}}_{\text{BL}} = \boldsymbol{\omega}$ in a certain environment y. For another environment that is twice as bright: $y' = 2y$, the resulting $\hat{\boldsymbol{\omega}}_{\text{BL}}$ will be off by a factor of 4. Nevertheless, we will show that $\hat{\boldsymbol{\omega}}_{\text{BL}}$ and $\hat{\boldsymbol{v}}_{\text{BL}}$ are useful for control purposes, since the unknown gain factor plays less role in system stability.

C. Velocity damping using bilinear velocity estimates

We will now prove the main result of this section: the control law constructed from the bilinear estimates will regulate the velocities to 0. This is actually not immediate, because the we have been treating rotational and translational motions separately, whereas the dynamics of $\boldsymbol{\omega}$ and $\boldsymbol{v}$ interact in practice. The condition for this to hold is again related to the modified contrast matrix. This is not too surprising because we have dropped the nearness $\mu(s)$ from the least-squares estimates like in the previous section.

Proposition 4 (Visual damping): Assume that $\widetilde{C}(\boldsymbol{q}, \alpha) > 0$. Then the control law

$$\begin{cases} \boldsymbol{\tau} = \ -k_d\, \hat{\boldsymbol{\omega}}_{\text{BL}} = -k_d\, \langle (\text{S}y) \dot{y} \rangle\,, \\ \boldsymbol{f} = -\alpha k_d\, \hat{\boldsymbol{v}}_{\text{BL}} = -\alpha k_d\, \langle (\nabla y) \dot{y} \rangle\,, \end{cases}$$

with $k_d > 0$, globally stabilizes $\boldsymbol{\omega}, \boldsymbol{v}$ to 0.

Proof: Take the kinetic energy

$$V(\boldsymbol{\omega}, \boldsymbol{v}) = \frac{1}{2} m \boldsymbol{v}^* \boldsymbol{v} + \frac{1}{2} \boldsymbol{\omega}^* \mathbb{I} \boldsymbol{\omega}$$

as a Lyapunov candidate, and compute the time derivative of V:

$$\begin{aligned} \dot{V} &= \boldsymbol{v}^* m \dot{\boldsymbol{v}} + \boldsymbol{\omega}^* \mathbb{I} \dot{\boldsymbol{\omega}} \\ &= \boldsymbol{v}^* (m \boldsymbol{v} \times \boldsymbol{\omega} + \boldsymbol{f}) + \boldsymbol{\omega}^* ((\mathbb{I}\boldsymbol{\omega}) \times \boldsymbol{\omega} + \boldsymbol{\tau}) \\ &= \boldsymbol{v}^* \boldsymbol{f} + \boldsymbol{\omega}^* \boldsymbol{\tau} \\ &= \boldsymbol{v}^* (-\alpha k_d \langle (\nabla y) \dot{y} \rangle) + \boldsymbol{\omega}^* (-k_d \langle (\text{S}y) \dot{y} \rangle). \end{aligned}$$

By the optic flow equation (4),

$$\begin{aligned} \langle (\nabla y) \dot{y} \rangle &= \langle \mu \nabla y \nabla y^* \rangle\, \boldsymbol{v} + \langle \nabla y \text{S} y^* \rangle\, \boldsymbol{\omega}, \\ \langle (\text{S}y) \dot{y} \rangle &= \langle \mu \text{S}y \nabla y^* \rangle\, \boldsymbol{v} + \langle \text{S}y \text{S}y^* \rangle\, \boldsymbol{\omega}. \end{aligned}$$

Thus $\dot{V}$ is a quadratic function of $(\boldsymbol{\omega}, \boldsymbol{v})$:

$$\begin{aligned} \dot{V} &= -k_d \left[\alpha \boldsymbol{v}^* \left(\langle \mu \nabla y \nabla y^* \rangle\, \boldsymbol{v} + \langle \nabla y \text{S} y^* \rangle\, \boldsymbol{\omega} \right) \right. \\ &\qquad \left. + \boldsymbol{\omega}^* \left(\langle \mu \text{S}y \nabla y^* \rangle\, \boldsymbol{v} + \langle \text{S}y \text{S}y^* \rangle\, \boldsymbol{\omega} \right) \right] \\ &= -k_d \boldsymbol{z}^* F \boldsymbol{z} \\ &= -k_d \boldsymbol{z}^* \widetilde{C} \boldsymbol{z}, \end{aligned}$$

where $\boldsymbol{z} = [\boldsymbol{\omega}^*\ \boldsymbol{v}^*]^*$. If the modified contrast matrix $\widetilde{C}$ is positive definite, then $\dot{V}$ is negative definite, and thus the system converges to $(\boldsymbol{\omega}, \boldsymbol{v}) = 0$ asymptotically. ∎

Note that the velocity control law in the previous section also shares the same particularly simple computational form, being bilinear in y and $(g - y)$. We will further discuss its consequences in later sections.

V. Visual control in force/torque

By combining the results from the previous two sections, we now state the main theoretical result of this paper, namely the visual proportional-derivative (PD) control of the full rigid-body dynamics, with force $\boldsymbol{f}$ and torque $\boldsymbol{\tau}$ as the control inputs. Aside from the condition on the modified contrast matrix $\widetilde{C}$, the only additional requirement to ensure stability is that the damping factor (i.e., k_d) is "large enough".

Proposition 5 (Proportional-derivative control): Assume that the modified contrast matrix $\widetilde{C}(\boldsymbol{q}, \alpha)$ is positive definite at $\boldsymbol{q} = \boldsymbol{q}_g$. Then the control law

$$\begin{cases} \boldsymbol{\tau} = \ \mathbb{I} \langle (\text{S}y)(g - y) \rangle\ - k_d \mathbb{I} \langle (\text{S}y)\, \dot{y} \rangle\,, \\ \boldsymbol{f} = \alpha m \langle (\nabla y)\, (g - y) \rangle - \alpha m k_d \langle (\nabla y)\, \dot{y} \rangle\,, \end{cases} \tag{13}$$

guarantees local asymptotic stability of the second-order system (2) near $\boldsymbol{q} = \boldsymbol{q}_g$ for large enough k_d.

Proof: Linearize the system (2) near $\boldsymbol{q} = \boldsymbol{q}_g$. By using the exponential coordinates $\boldsymbol{\varphi}$, the system has the form (see Appendix A):

$$\begin{cases} \dot{\boldsymbol{\xi}} = \boldsymbol{z}, \\ \dot{\boldsymbol{z}} = -F \boldsymbol{\xi} - k_d F \boldsymbol{z}, \end{cases}$$

where $\boldsymbol{\xi} = [\boldsymbol{\varphi}^*\ \boldsymbol{p}^*]^*$ and $\boldsymbol{z} = [\boldsymbol{\omega}^*\ \boldsymbol{v}^*]^*$. We can prove stability of this system by considering a Lyapunov candidate

$$\begin{aligned} V &= \boldsymbol{\xi}^* \widetilde{C} \boldsymbol{\xi} + \frac{1}{2} \boldsymbol{z}^* \boldsymbol{z} + k_d^{-1} \boldsymbol{\xi}^* \boldsymbol{z} \\ &= \begin{bmatrix} \boldsymbol{\xi} \\ \boldsymbol{z} \end{bmatrix}^* \begin{bmatrix} \frac{1}{2}(F + F^*) & \frac{1}{2} k_d^{-1} I \\ \frac{1}{2} k_d^{-1} I & \frac{1}{2} I \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi} \\ \boldsymbol{z} \end{bmatrix}. \end{aligned}$$

Because $F + F^* = 2\widetilde{C}(\boldsymbol{q}_g, \alpha) > 0$, the above function is positive definite for large enough k_d. We also compute the derivative of V as follows:

$$\begin{aligned}\dot{V} &= \boldsymbol{\xi}^* F \dot{\boldsymbol{\xi}} + \dot{\boldsymbol{\xi}}^* F \boldsymbol{\xi} + \boldsymbol{z}^* \dot{\boldsymbol{z}} + k_d^{-1} \boldsymbol{\xi}^* \dot{\boldsymbol{z}} + k_d^{-1} \dot{\boldsymbol{\xi}}^* \boldsymbol{z} \\ &= \boldsymbol{\xi}^* F \boldsymbol{z} + \boldsymbol{z}^* F \boldsymbol{\xi} + \boldsymbol{z}^* (-F\boldsymbol{\xi} - k_d F \boldsymbol{z}) \\ &\quad + k_d^{-1} \boldsymbol{\xi}^* (-F\boldsymbol{\xi} - k_d F \boldsymbol{z}) + k_d^{-1} \boldsymbol{z}^* \boldsymbol{z} \\ &= -\boldsymbol{z}^* (k_d F - k_d^{-1} I) \boldsymbol{z} - \boldsymbol{\xi}^* k_d^{-1} F \boldsymbol{\xi} \\ &= -\boldsymbol{z}^* (k_d \widetilde{C}(\boldsymbol{q}_g, \alpha) - k_d^{-1} I) \boldsymbol{z} - \boldsymbol{\xi}^* k_d^{-1} \widetilde{C}(\boldsymbol{q}_g, \alpha) \boldsymbol{\xi}.\end{aligned}$$

If k_d is large enough, then $(k_d \widetilde{C}(\boldsymbol{q}_g, \alpha) - k_d^{-1} I) > 0$, and $\dot{V}$ is negative definite. Thus the linearized system is asymptotically stable, and the original system is locally asymptotically stable. ■

Aside from various extra gain factors, the above control law is a direct combination of the control laws in velocity ("proportional" part) and the velocity damping laws ("derivative" part) from Proposition 2 and 4, respectively. Note that the derivative part is not the same as taking the time derivative of the proportional part, since the two are derived differently: the proportional part comes from the gradient of the quadratic metric over the images; the derivative part comes from the bilinear approximation of the velocity estimates.

VI. Computational structure

So far we have been assuming a spatially continuous visual sensor to derive the theoretical results. However, since physical visual sensors (cameras, fly's compound eye) have only a finite number of photodetectors, the PD control law needs to be discretized before being implemented. This will also give insights on the computational structure of the control law, especially its *localness* (which implies sparsity) in computation.

Recall that the directions of pixels are denoted as $s_i \in \mathbb{S}^2$, and define the discretized visual input as $\mathbf{g}_i \triangleq g(s_i)$ and $\mathbf{y}_i \triangleq y(s_i)$. Now $\mathbf{g}$ and $\mathbf{y}$ are vectors in $\mathbb{R}^n$, where n is the total number of pixels. We also need to discretize the differential operators S and ∇ to approximate the result of differentiation at each pixel locations for an arbitrary input image y. In an informal notation, we look for tensors $\mathbf{A}$ and $\mathbf{B}$ that act on discretized images and satisfy:

$$\text{Discretize}\,[\mathrm{S}y] \approx \mathbf{A}\mathbf{y}, \qquad \text{Discretize}\,[\nabla y] \approx \mathbf{B}\mathbf{y}.$$

Arithmetically, both $\mathbf{A}$ and $\mathbf{B}$ can be thought as $n \times n \times 3$ arrays: in the above equations, they act on vectors in $\mathbb{R}^n$ and return $n \times 3$ matrices. We will use the notation $\mathbf{A}_{ijk}$ to denote the entries of a tensor $\mathbf{A}$, and a missing subscript to imply the collection of all the entries in the corresponding dimension (like the operation ":" on array indices in MATLAB/Octave). For example, $\mathbf{A}_{ij}$ represents a vector in $\mathbb{R}^3$.

One choice of approximation is to apply spatial smoothing before differentiation, akin to the Sobel operator used in computer vision for edge detection. The difference is that here the approximated differentiation does not assume a uniformly sampled visual field, and can be applied to an arbitrary disposition of pixels. Under this approximation, the tensors $\mathbf{A}$ and $\mathbf{B}$ can be obtained as (up to normalization factors, see Appendix B for detail):

$$\begin{aligned}\mathbf{A}_{ij} &= -q'(\alpha_{ij}) \cdot (s_i \times s_j) / \sin(\alpha_{ij}), \\ \mathbf{B}_{ij} &= -q'(\alpha_{ij}) \cdot (\mathbf{I} - s_i s_i^*) s_j / \sin(\alpha_{ij}),\end{aligned} \tag{14}$$

where $\mathbf{I}$ is the 3×3 identity matrix, q is a smooth kernel function (e.g., Gaussian), q' denotes its derivative, and $\alpha_{ij} \triangleq \cos^{-1}(s_i \cdot s_j)$ is the angle formed by s_i and s_j. We adopt the convention that $\mathbf{A}_{ii} = \mathbf{B}_{ii} = 0$. Note that, if chosen appropriately, the kernel q can be localized and endows a *sparse approximation* for $\mathbf{A}$ and $\mathbf{B}$. This important feature makes the method numerically favorable during implementation, despite that the number of entries in $\mathbf{A}$ and $\mathbf{B}$ could be large for a practical sensor. Moreover, the *localness* of the kernel also implies that this method could have an efficient implementation on existing parallel computational architectures (e.g., GPU), where access to non-local data is expensive. More discussions on the bio-plausibility of such computation will be given later in Section VII.

Under the new (discrete) notation, the previous PD control law (13) becomes:

$$\begin{cases} \boldsymbol{\tau}_k = \sum_{i,j} \mathbf{y}_i \mathbf{A}_{ijk} \left[(\mathbf{g}_j - \mathbf{y}_j) - k_d \dot{\mathbf{y}}_j \right], \\ \boldsymbol{f}_k = \sum_{i,j} \mathbf{y}_i \mathbf{B}_{ijk} \left[(\mathbf{g}_j - \mathbf{y}_j) - k_d \dot{\mathbf{y}}_j \right]. \end{cases} \tag{15}$$

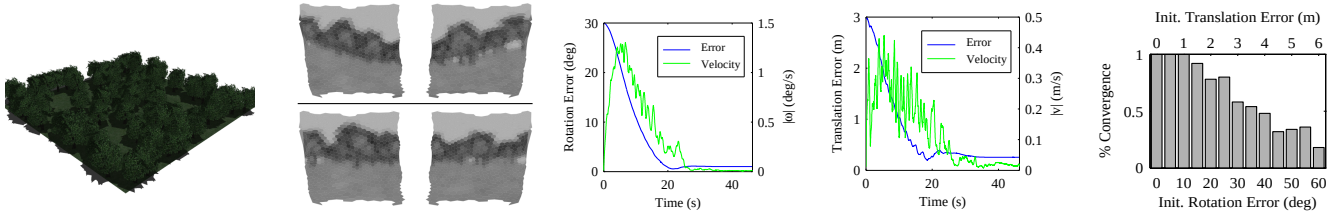
For conciseness, we have ignored some constant gain factors such as $\mathbb{I}$ and m and collected the two parts in (13). The index k spans the three components of force and torque; the indices i and j range from 1 to n: $1 \le i, j \le n$. More concretely, in MATLAB/Octave, synthesis of the discretized control law (15) will look like the following:

```
% n: total number of pixels
% For perspective cameras, n = width * height
g = reshape(goal_image, n, 1);
y = reshape(current_image, n, 1);
for k = 1:3
    torque(k) = y' * A(:,:,k) * (g - y - kd * y_dot);
    force(k)  = y' * B(:,:,k) * (g - y - kd * y_dot);
end
```

This shows that the computation is simple and robust, compared with traditional point-feature-based methods that often involve feature extraction, matching, and outlier removal, etc.

VII. Bio-plausiblity of the computation

The control law introduced in Section V can be considered bio-plausible because of its simple computational structure. It is a multilinear form of y, g, $\dot{y}$, combined in a simple and feedforward way, thus can be potentially realized on a neural substrate [3]. In synthesis, the computation is a wide-field integration of local nonlinear operations. In [14], we showed that part of the computation is equivalent to a wide-field integration of elementary motion detectors, which is an accepted model for capturing many aspects of the visual computation in the fruit fly brain [20]. More generally, the visual inputs y and g do not need be the raw luminance as observed directly. As an example, it has been widely accepted that flies perform contrast normalization on the perceived



(a) CAD model of a natural environment (b) Start (top) and goal (bottom) images (c) Rotation error and $|\boldsymbol{\omega}|$ (d) Translation error and $|\boldsymbol{v}|$ (e) Convergence test

Figure 4. Simulation results using a natural environment as in (a). The initial current image and the goal image are shown in (b) using Mercator projection. We visualize the rotation error using the geodesic distance on $\mathrm{SO}(3)$ [16]. In all simulations, we determined the length scale to ensure that the features have reasonable realistic size (e.g. the height of a tree is 10–15 m), and the time scale to obtain physically achievable kinematics (e.g. the fruit fly has a typical translational velocity of 0.2 m/s and maximum of 1.2 m/s).

luminance during its early stage of visual processing [21]. Such operation is considered bio-plausible as long as they still return some form of "dense" visual input. To corroborate the bio-plausibility, we will show simulation results of the control law for flight stabilization of a fruit fly.

We have used *fsee* [22] to simulate the visual input y of the fruit fly *Drosophila*. It takes a 3D CAD world model (Fig. 4a) and renders the visual perception as per the compound-eye optics of the fruit fly. The simulation incorporates the spatial disposition of the 1,398 "pixels" (ommatidia) that form the compound eye, spatial blurring, and temporal filtering of the underlying photoreceptors, all based on current knowledge in visual physiology. Fig. 4b shows a typical simulation of the visual stimulus in Mercator projection. The temporal change in y is approximated using the first-order difference: $\dot{y}(t) = \frac{1}{\Delta t}\,[\,y(t) - y(t - \Delta t)]$. At each time step, the pose of the fly is updated by discretizing the rigid-body dynamics (2). The visual input is then updated accordingly using *fsee* from the new pose. We have ignored the detailed dynamics (wing, multi-body, etc.) and made a simplified assumption that the dynamics of the fruit fly can be treated as a rigid-body. During simulation, all the physical quantities are scaled based on experimental data from fruit flies: for example, the typical translational velocity of a fruit fly is around 0.2 m/s.

A typical convergent example using the proportional-derivative control law (13) is shown in Fig. 4c (rotation component) and Fig. 4d (translation component). The simulation starts at a joint rotational error of 30 deg and translational error of 3 m. Eventually, the rotation error settles down to about 1 deg, which is reasonable given the spatial discretization of the sensor: each "pixel" spans about 5 steradians. This nonzero rotation error induces a corresponding small nonzero translation error. The nonzero, in fact oscillating, final velocities are probably due to numerical discretization in the simulation. Since the control law is only guaranteed to be locally stable, we also used the Monte Carlo method to perform numerical tests of the region of attraction by sampling from a large number of initial conditions. The percentage of convergent cases is shown in Fig. 4e for increasing, randomly sampled initial (joint) rotation and translation error. Despite the fact that the environment has few distinctive features, the control law provides a reasonable region of attraction, since it does not rely on feature extraction.

VIII. Experiments

To demonstrate the engineering value of the bilinear control law, we have also implemented the control law on an indoor helicopter testbed. The helicopter is customized from the E-flite Blade CX2 coaxial indoor helicopter, by adding wireless camera(s) onboard as the vision sensor (Fig. 5). Due to limited payload, the helicopter is only able to carry 1-2 cameras, which do not give an omnidirectional view of the environment. All the control commands computed by the PC are sent to the radio controller (Spektrum DX6i) via its trainer port after being converted to PPM format by a customized microcontroller (Endurance PCTx). The wireless camera placed on the helicopter (Swann MicroCam) is able to provide 640×480 images at a maximum frame rate of 30 fps. Due to computational constraints, we choose to subsample the relayed camera images at 220×166. To compensate for lighting variations, the images are passed though a Sobel edge-detection filter followed by thresholding and Gaussian smoothing ($\sigma = 15$ px) before being used to compute the control law. The control laws are computed using the approximated discrete tensors (14). The majority of the computation is performed on an Intel Core2 Duo 2.40 GHz machine. The total delay of the entire control procedure is estimated to be 78 ms.

The helicopter has 4 control inputs: throttle, rudder, aileron, and elevation. Since the pitch of the rotor blades is fixed, the throttle command u_{thr} controls the thrust by changing the speeds of the upper and lower rotors collectively. Yaw control is realized by tuning the differential speed between the two counter-rotating rotors via the rudder command u_{rud}. Lateral movements are controlled by the aileron and elevation commands u_{ail} and u_{ele}, which determine the cyclic pitch of the lower rotor blades by driving a 2-DOF swashplate. One limitation of our helicopter system is that it is underactuated, because it has 6 DOFs with only 4 control inputs. This is evident in that the helicopter must, for example, pitch forward in order to initiate forward flight. In principle, a different model rather than (2) needs to be adopted to take into account the underactuated dynamics. However, when the helicopter is near hover, where the uncontrolled pitching and rolling are small, the helicopter dynamics can be well approximated by

a simplified fully actuated model with 4 control inputs. By using Euler angles (pitch θ, roll ψ, and yaw ϕ) as the local coordinates for rotation near $\boldsymbol{r}_g = \mathrm{Id}$, the linearized simplified model becomes:

$$\mathbb{I}_z \ddot{\phi} = u_{\text{rud}} + b_{\text{rud}},$$
$$m\ddot{\boldsymbol{p}} = \begin{bmatrix} u_{\text{ail}} \\ u_{\text{ele}} \\ u_{\text{thr}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ b_{\text{thr}} + f_{\text{gravity}} \end{bmatrix}.$$

We have also included the hardware trim/bias in throttle and rudder, namely b_{thr} and b_{rud}. In practice, these can often depend on various unknown factors such the remaining battery capacity, battery placement, and motor temperature. Moreover, the change in throttle trim will also affect other channels (rudder, aileron, elevation) because the lateral thrust is provided via the main rotors.

Compared to simulations, the testbed now introduces several technical challenges due to various non-idealities: 1) limited field of view, 2) underactuated dynamics, 3) delay in the control loop, 4) non-Gaussian sensor noise due to wireless interference, and 5) time-varying hardware uncertainties.

Automatic bias compensation: The first problem in operating such a helicopter is compensating the biases that make the helicopter drift away when one applies nominal inputs. Bias compensation is usually a long trial-and-error manual operation that must be often repeated. However, we discovered that the bilinear velocity estimates $\hat{\boldsymbol{\omega}}_{\text{BL}}$ and $\hat{\boldsymbol{v}}_{\text{BL}}$ defined in (12) were so reliable that they provided an opportunity to perform simple and effective automatic bias compensation. For example, if $\hat{\boldsymbol{\omega}}_{\text{BL},z} > 0$, we know, even if the magnitude is unknown, that the helicopter is rotating to the left, and we can compensate by decreasing the rudder command. In formulas, we set

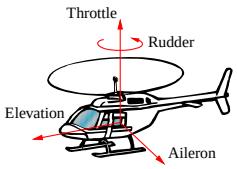
$$u_{\text{thr}}(t) \propto \int_0^t \eta(\tau)\hat{\boldsymbol{v}}_{\text{BL},z}(\tau)\,d\tau, \quad u_{\text{rud}}(t) \propto \int_0^t \eta(\tau)\hat{\boldsymbol{\omega}}_{\text{BL},z}(\tau)\,d\tau.$$

In these expressions, $\eta(\tau) = e^{-\gamma\tau}$ is a decay factor used to suppress undesirable oscillations due to noise. As can be seen from Fig. 6b, the oscillations in rudder compensation (shown in Fig. 6a) disappear after a decay factor is used.

Visual pose stabilization: An attached video[1] shows a demo of visual pose stabilization. Currently, we keep the throttle manually controlled for safety purpose, but leave the remaining three channels automatically controlled. Fig. 7a and 7b show the goal image and a typical current image; both the unprocessed one as seen from the camera and the processed one after Sobel edge detection and Gaussian smoothing are shown. Because we do not know the ground truth of the current pose, the only useful indicator is the cost function J defined in (6), whose change over time is plotted in Fig. 7c. The oscillations in J and the fact that J is close to 0 during the first ~ 30 s of the trial indicate that the helicopter is being stabilized around the goal.

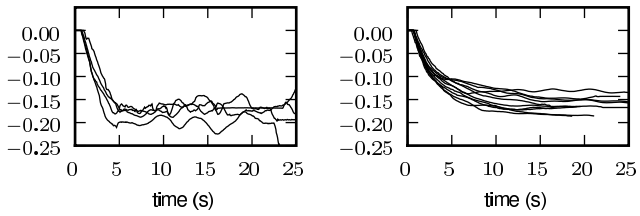
[1]A high-quality one is also available at: http://purl.org/hanshuo/2010/pd_pose_stabilization

(a) Customized helicopter with wireless cameras



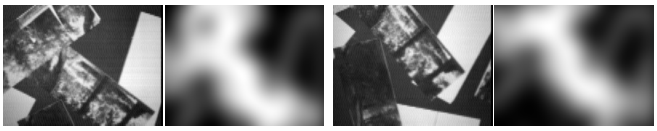
(b) Illustration of the control inputs

Figure 5. (a) Helicopter used in the experiments, custom-built from E-flite Blade CX2. One wireless camera in encased in the tail boom. (b) Illustration of the control inputs: throttle u_{thr}, rudder u_{rud}, aileron u_{ail}, and elevation u_{ele}.

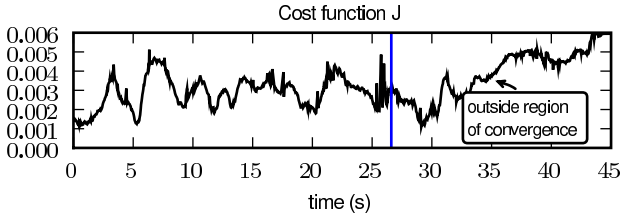


(a) Without decay factor γ (b) With decay factor γ

Figure 6. Automatic trim/bias compensation for rudder (rotation about the z-axis) using visual input only, recorded from a number of repeated trials. The compensation term is synthesized by integrating the bilinear velocity estimates $\hat{\boldsymbol{\omega}}_{\text{BL}}$ and $\hat{\boldsymbol{v}}_{\text{BL}}$ over time. Using a decay factor suppresses the oscillations and gives a stable compensation.



(a) The goal image (before and after processing). (b) Image at time $t \approx 26$ s (before and after processing).



(c) Evolution in time of the cost function J. The blue line indicates the time when the image in (b) was taken.

Figure 7. Pose stabilization using the visual PD control law. (a) The goal image and an example current image used during the test flight. The processing includes Sobel edge detection and subsequent Gaussian smoothing. (b) The change of the cost function J in time during the test flight, as an indication of stabilization around the goal. At the end of the trial, due to several non-idealities such camera noise and drifts in the hardware trims, the helicopter went out of the region of attraction.

IX. Conclusions

We showed that the task of visual servoing can be solved (at least locally) through purely visual control laws. The non-obvious result is that, a locally stable control law can be obtained without knowing the distance to the objects, given that certain condition on the environment holds. These control laws respect the constraints of bio-plausible computation by using a bilinear/quadratic operation on the raw luminance: in

a traditional implementation, the operation is only a tensor-vector multiplication with the raw pixel array. We are in the process of extending our work in two directions. From the engineering viewpoint, we want to ascertain whether the approach can give solid advantages over traditional methods (feature-based and not). This would imply looking at issues such as change of lighting condition, occlusions, locality, nonholonomic constraints, which we did not consider yet. At a more abstract level, we want to investigate whether the same bio-plausible design concept is applicable for more complex tasks, such as structure from motion and obstacle avoidance.

Acknowledgements. This work is supported in part by the US Army Institute for Collaborative Biotechnology (ICB) and the Boeing Corporation.

APPENDIX A
LINEARIZATION OF THE SYSTEM KINEMATICS/DYNAMICS

Linearization of the systems in Proposition 1, 2, and 5 will use the "directional derivatives" (formally, tangent maps) of the visual input y with respect to the pose. The results are closely related to the differential operators S and ∇, and we state them here as a lemma. The main technique used for deriving the following results is calculus on the manifold $\mathrm{SE}(3)$. We will use the exponential coordinates $\boldsymbol{\varphi}$ to represent rotation, since we are only interested the local properties near $\boldsymbol{r}_g = \mathrm{Id}$.

Lemma 2: Near $\boldsymbol{q} = \mathrm{Id}$, or $(\boldsymbol{\varphi}, \boldsymbol{p}) = (0, 0)$, the tangent maps ("directional derivatives") of the visual input y with respect to the pose are

$$D_{\boldsymbol{\varphi}} y|_{\boldsymbol{q}=\mathrm{Id}} \cdot \boldsymbol{\omega} = (\mathrm{S}y)^* \boldsymbol{\omega},$$
$$D_{\boldsymbol{p}} y|_{\boldsymbol{q}=\mathrm{Id}} \cdot \boldsymbol{v} = \mu(\nabla y)^* \boldsymbol{v}.$$

Proof: By the chain rule and the system kinematics (1),

$$\dot{y} = D_{\boldsymbol{p}} y \cdot \dot{\boldsymbol{p}} + D_{\boldsymbol{r}} y \cdot \dot{\boldsymbol{r}} = D_{\boldsymbol{p}} y \cdot (\boldsymbol{r}\boldsymbol{v}) + D_{\boldsymbol{r}} y \cdot \left[\boldsymbol{r}\,(\boldsymbol{\omega})^\wedge\right].$$

Comparing with the optic flow equation (4) near $\boldsymbol{q} = \mathrm{Id}$, we have

$$D_{\boldsymbol{r}} y|_{\boldsymbol{q}=\mathrm{Id}} \cdot \boldsymbol{\omega}^\wedge = (\mathrm{S}y)^* \boldsymbol{\omega},$$
$$D_{\boldsymbol{p}} y|_{\boldsymbol{q}=\mathrm{Id}} \cdot \boldsymbol{v} = \mu(\nabla y)^* \boldsymbol{v}.$$

By the definition of the exponential coordinates: $\boldsymbol{r} = \mathrm{Exp}(\boldsymbol{\varphi}^\wedge)$ and the chain rule,

$$D_{\boldsymbol{\varphi}}(y(\boldsymbol{r}))|_{\boldsymbol{\varphi}=0} \cdot \boldsymbol{\omega} = D_{\boldsymbol{r}}(y(\boldsymbol{r}))|_{\boldsymbol{r}=\mathrm{Id}} \circ D_{\boldsymbol{\varphi}}\mathrm{Exp}(\boldsymbol{\varphi}^\wedge)|_{\boldsymbol{\varphi}=0} \cdot \boldsymbol{\omega}.$$

Here we have omitted the evaluation at $\boldsymbol{p} = 0$ on both sides for conciseness. Note that for any matrix A, $D_A \mathrm{Exp}(A)|_{A=\mathrm{Id}} = \mathrm{Id}$, and the hat map $\wedge$ is linear: $D_{\boldsymbol{\varphi}}(\boldsymbol{\varphi}^\wedge) \cdot u = u^\wedge$. We then obtain the final results:

$$D_{\boldsymbol{\varphi}} y|_{\boldsymbol{\varphi}=0} \cdot \boldsymbol{\omega} = D_{\boldsymbol{r}}(y(\boldsymbol{r}))|_{\boldsymbol{r}=\mathrm{Id}} \cdot \boldsymbol{\omega}^\wedge = (\mathrm{S}y)^* \boldsymbol{\omega}.$$

or written in full,

$$D_{\boldsymbol{\varphi}} y|_{\boldsymbol{q}=\mathrm{Id}} \cdot \boldsymbol{\omega} = (\mathrm{S}y)^* \boldsymbol{\omega}.$$

■

Now we are ready to derive the linearization of the system in Proposition 1.

Lemma 3: The linearization of the system (9) near $(\boldsymbol{\varphi}, \boldsymbol{p}) = (0, 0)$ with the exact gradient control law (7) is

$$\dot{\boldsymbol{\xi}} = -C(\boldsymbol{q}_g)\boldsymbol{\xi}, \qquad \boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\varphi} \\ \boldsymbol{p} \end{bmatrix},$$

where C is the contrast matrix defined in Proposition 1.

Proof: The linearization around the equilibrium $(\boldsymbol{\varphi}, \boldsymbol{p}) = (0, 0)$ is

$$\begin{cases} \dot{\boldsymbol{\varphi}} = D_{\boldsymbol{\varphi}}(\boldsymbol{\omega}) \cdot \boldsymbol{\varphi} + D_{\boldsymbol{p}}(\boldsymbol{\omega}) \cdot \boldsymbol{p}, \\ \dot{\boldsymbol{p}} = D_{\boldsymbol{\varphi}}(\boldsymbol{v}) \cdot \boldsymbol{\varphi} + D_{\boldsymbol{p}}(\boldsymbol{v}) \cdot \boldsymbol{p}, \end{cases} \tag{16}$$

where all the derivatives are evaluated at $(\boldsymbol{\varphi}, \boldsymbol{p}) = (0, 0)$, or $\boldsymbol{q} = \mathrm{Id}$:

$$\begin{aligned} D_{\boldsymbol{\varphi}}(\boldsymbol{\omega})|_{\boldsymbol{q}=\mathrm{Id}} &= D_{\boldsymbol{\varphi}}(\langle(\mathrm{S}y)(g-y)\rangle)|_{\boldsymbol{q}=\mathrm{Id}} \\ &= -\langle(\mathrm{S}y)(D_{\boldsymbol{\varphi}} y)\rangle\,|_{\boldsymbol{q}=\mathrm{Id}} = -\langle(\mathrm{S}y)(\mathrm{S}y)^*\rangle, \end{aligned}$$

$$\begin{aligned} D_{\boldsymbol{p}}(\boldsymbol{\omega})|_{\boldsymbol{q}=\mathrm{Id}} &= D_{\boldsymbol{p}}(\langle(\mathrm{S}y)(g-y)\rangle)|_{\boldsymbol{q}=\mathrm{Id}} \\ &= -\langle(\mathrm{S}y)(D_{\boldsymbol{p}} y)\rangle\,|_{\boldsymbol{q}=\mathrm{Id}} = -\langle\mu(\mathrm{S}y)(\nabla y)^*\rangle, \end{aligned}$$

$$\begin{aligned} D_{\boldsymbol{\varphi}}(\boldsymbol{v})|_{\boldsymbol{q}=\mathrm{Id}} &= D_{\boldsymbol{\varphi}}(\langle\mu(\nabla y)(g-y)\rangle)|_{\boldsymbol{q}=\mathrm{Id}} \\ &= -\langle(\nabla y)(D_{\boldsymbol{\varphi}} y)\rangle\,|_{\boldsymbol{q}=\mathrm{Id}} = -\langle\mu(\nabla y)(\mathrm{S}y)^*\rangle, \end{aligned}$$

$$\begin{aligned} D_{\boldsymbol{p}}(\boldsymbol{v})|_{\boldsymbol{q}=\mathrm{Id}} &= D_{\boldsymbol{p}}(\langle\mu(\nabla y)(g-y)\rangle)|_{\boldsymbol{q}=\mathrm{Id}} \\ &= -\langle\mu(\nabla y)(D_{\boldsymbol{p}} y)\rangle\,|_{\boldsymbol{q}=\mathrm{Id}} = -\left\langle\mu^2(\nabla y)(\nabla y)^*\right\rangle. \end{aligned}$$

In the above, we have used the fact that g does not depend on $\boldsymbol{q}$, and $g = y$ at $\boldsymbol{q} = \mathrm{Id}$. The lemma then follows by substituting the above results into (16). ■

The linearization of the system in Proposition 2 can be proved in a similar approach.

Lemma 4: The linearization of the system (9) near $(\boldsymbol{\varphi}, \boldsymbol{p}) = (0, 0)$ with the approximate gradient control law (10) is

$$\dot{\boldsymbol{\xi}} = -F\boldsymbol{\xi}, \qquad \boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\varphi} \\ \boldsymbol{p} \end{bmatrix},$$

where F is the matrix defined in the proof of Proposition 2.

Proof: Because the control law for $\boldsymbol{\omega}$ remains the same, it suffices to reevaluate the derivatives of $\boldsymbol{v}$ near $\boldsymbol{q} = \mathrm{Id}$:

$$\begin{aligned} D_{\boldsymbol{\varphi}}(\boldsymbol{v})|_{\boldsymbol{q}=\mathrm{Id}} &= D_{\boldsymbol{\varphi}}(\alpha\,\langle(\nabla y)(g-y)\rangle)|_{\boldsymbol{q}=\mathrm{Id}} \\ &= -\langle(\nabla y)(D_{\boldsymbol{\varphi}} y)\rangle\,|_{\boldsymbol{q}=\mathrm{Id}} = -\langle\alpha(\nabla y)(\mathrm{S}y)^*\rangle, \end{aligned}$$

$$\begin{aligned} D_{\boldsymbol{p}}(\boldsymbol{v})|_{\boldsymbol{q}=\mathrm{Id}} &= D_{\boldsymbol{p}}(\alpha\,\langle(\nabla y)(g-y)\rangle)|_{\boldsymbol{q}=\mathrm{Id}} \\ &= -\alpha\,\langle(\nabla y)(D_{\boldsymbol{p}} y)\rangle\,|_{\boldsymbol{q}=\mathrm{Id}} = -\alpha\,\langle\mu(\nabla y)(\nabla y)^*\rangle. \end{aligned}$$

The lemma follows by substituting the above and the previous results in Lemma 3 for $\boldsymbol{\omega}$ into (16). ■

Lastly, we will show the linearization of the system dynamics in Proposition 5.

Lemma 5: In exponential coordinates $(\boldsymbol{\varphi}, \boldsymbol{\omega}, \boldsymbol{p}, \boldsymbol{v})$, the linearization of the system (2) under the control law (13) near the equilibrium $(0, 0, 0, 0)$ is

$$\begin{cases} \dot{\boldsymbol{\xi}} = \boldsymbol{z}, \\ \dot{\boldsymbol{z}} = -F\boldsymbol{\xi} - k_d F\boldsymbol{z}, \end{cases}$$

where $\boldsymbol{\xi} = [\boldsymbol{\varphi}^* \ \boldsymbol{p}^*]^*$ and $\boldsymbol{z} = [\boldsymbol{\omega}^* \ \boldsymbol{v}^*]^*$.

Proof: The linearization around the equilibrium has the following form:

$$\begin{cases} \dot{\boldsymbol{\varphi}} &= \boldsymbol{\omega}, \\ \mathbb{I}\dot{\boldsymbol{\omega}} &= 0 + D_{\boldsymbol{\varphi}}\boldsymbol{\tau}\cdot\boldsymbol{\varphi} + D_{\boldsymbol{p}}\boldsymbol{\tau}\cdot\boldsymbol{p} + D_{\boldsymbol{\omega}}\boldsymbol{\tau}\cdot\boldsymbol{\omega} + D_{\boldsymbol{v}}\boldsymbol{\tau}\cdot\boldsymbol{v}, \\ \dot{\boldsymbol{p}} &= \boldsymbol{v}, \\ m\dot{\boldsymbol{v}} &= 0 + D_{\boldsymbol{\varphi}}\boldsymbol{f}\cdot\boldsymbol{\varphi} + D_{\boldsymbol{p}}\boldsymbol{f}\cdot\boldsymbol{p} + D_{\boldsymbol{\omega}}\boldsymbol{f}\cdot\boldsymbol{\omega} + D_{\boldsymbol{v}}\boldsymbol{f}\cdot\boldsymbol{v}. \end{cases} \tag{17}$$

where all the derivatives are computed at the equilibrium, and the control law is given by (13):

$$\begin{cases} \boldsymbol{\tau} = \ \mathbb{I}\left\langle(\mathrm{S}y)(g-y)\right\rangle - k_d\mathbb{I}\left\langle(\mathrm{S}y)\,\dot{y}\right\rangle, \\ \boldsymbol{f} = \alpha m\left\langle(\nabla y)\,(g-y)\right\rangle - \alpha m k_d\left\langle(\nabla y)\,\dot{y}\right\rangle. \end{cases}$$

Note that some of the derivatives will vanish at the equilibrium $(\boldsymbol{\varphi}, \boldsymbol{\omega}, \boldsymbol{p}, \boldsymbol{v}) = (0,0,0,0)$:

$$\begin{aligned} &D_{\boldsymbol{\omega}}(\langle(\mathrm{S}y)(g-y)\rangle) = 0, \quad D_{\boldsymbol{\omega}}(\langle(\nabla y)(g-y)\rangle) = 0, \\ &D_{\boldsymbol{v}}(\langle(\mathrm{S}y)(g-y)\rangle) = 0, \quad D_{\boldsymbol{v}}(\langle(\nabla y)(g-y)\rangle) = 0, \\ &D_{\boldsymbol{\varphi}}(\langle(\mathrm{S}y)\,\dot{y}\rangle) = 0, \quad D_{\boldsymbol{\varphi}}(\langle(\nabla y)\,\dot{y}\rangle) = 0, \\ &D_{\boldsymbol{p}}(\langle(\mathrm{S}y)\,\dot{y}\rangle) = 0, \quad D_{\boldsymbol{p}}(\langle(\nabla y)\,\dot{y}\rangle) = 0. \end{aligned}$$

Substitute them into (17), and use the results in the proofs of Lemma 3 to obtain the thesis. ∎

APPENDIX B
APPROXIMATION OF THE DIFFERENTIAL OPERATORS

The original control law is written as a function of $\mathrm{S}y$ and ∇y. Here $y = y(s)$ is a function defined on the sphere, and $\mathrm{S}y$ and ∇y are vector fields defined on the sphere. In practice, one has only a discrete sensor that returns a discrete set of observations $\mathbf{y} \in \mathbb{R}^n$. The goal of approximation is to seek $\mathbf{A}$ and $\mathbf{B}$ that satisfy:

$$(\mathrm{S}y)\,|_{s=s_i} \approx [\mathbf{A}\mathbf{y}]_i\,, \qquad (\nabla y)\,|_{s=s_i} \approx [\mathbf{B}\mathbf{y}]_i\,.$$

Note that $\mathbf{A}\mathbf{y}$, $\mathbf{B}\mathbf{y}$ are discretized vector fields, with dimensions $n \times 3$. It follows that $\mathbf{A}, \mathbf{B}$ are tensors of dimension $n \times n \times 3$.

Discretization/approximation of a linear operator M is easy if M has a kernel realization; this means the result of M acting on any function y in $\mathbb{S}^2$, namely $\mathrm{M}y$, can be written in the following integral form:

$$(\mathrm{M}y)\,(s) = \int_{u\in\mathbb{S}^2} m(s,u)y(u)\,d\mathcal{S}, \tag{18}$$

where $m(s,u)$ is defined as the kernel of M. Given such M, it is straightforward to define its discretized version $\mathbf{M}$:

$$\mathbf{M}_{ij} \triangleq \rho(s_j)m(s_i,s_j). \tag{19}$$

Here, $\rho(s_j)$ is a normalization factor that depends on the sampled directions $\{s_j\}_{j=1}^n$. In the case of uniform sampling, $\rho(s_i)$ becomes a constant. Under this definition of $\mathbf{M}$, we have

$$\begin{aligned} \mathrm{M}y|_{s=s_i} &= \int_{u\in\mathbb{S}^2} m(s_i,u)y(u)\,d\mathcal{S} \\ &\approx \sum_j \rho(s_j)m(s_i,s_j)y(s_j) = \sum_j \mathbf{M}_{ij}\mathbf{y}_j = [\mathbf{M}\mathbf{y}]_i\,. \end{aligned}$$

Again, we see that the normalization factor $\rho(s_j)$ accounts for the approximation of the infinitesimal measure $d\mathcal{S}$ to compensate for any potential non-uniform sampling. The above result implies that the value of the continuous function $\mathrm{M}y$, when evaluated at s_i, can be approximated by the i-th entry of the discrete vector $\mathbf{M}\mathbf{y}$.

There are many ways to obtain or approximate the normalization $\rho(s)$ from a given configuration $\{s_j\}_{j=1}^n$. For example, one can use the area of the cell containing s_j in the Voronoi decomposition of $\mathbb{S}^2$. Starting from now, we will ignore $\rho(s)$ in all the following derivations for conciseness.

Unfortunately, the differential operators S and ∇ do not have a kernel realization. However, we will show that if we apply a smoothing operator Q before differentiation, the resulting composite operator will have a corresponding kernel. The smoothing operator Q is defined as

$$(\mathrm{Q}y)\,(s) = \int_{u\in\mathbb{S}^2} q(\alpha(s,u))y(u)\,d\mathcal{S}, \tag{20}$$

where $\alpha(s,u) = \cos^{-1}(s\cdot u)$ is the angle formed by the vectors s and u; $q(\cdot)$ is any smooth function in $\mathbb{R}$. A common choice would be the Gaussian function

$$q(\alpha) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{\alpha^2}{2\sigma^2}\right).$$

Give such a smoothing operator Q, we are able to define the composite operator to denote the smoothed differentiation and consider them as approximations of S and ∇

$$\mathrm{A} \triangleq \mathrm{S}\circ\mathrm{Q}, \qquad \mathrm{B} \triangleq \nabla\circ\mathrm{Q}.$$

Now we will show that both A and B adopt kernel realizations $a(s,u)$ and $b(s,u)$, respectively, and the discrete tensors $\mathbf{A}$ and $\mathbf{B}$ defined as (ignoring the normalization $\rho(s)$)

$$\mathbf{A}_{ij} \triangleq a(s_i,s_j), \qquad \mathbf{B}_{ij} \triangleq b(s_i,s_j), \tag{21}$$

have the form in (14).

Proposition 6: The tensors $\mathbf{A}$ and $\mathbf{B}$ defined in (21) have the form:

$$\begin{aligned} \mathbf{A}_{ij} &= -q'(\alpha_{ij})\cdot(s_i\times s_j)/\sin(\alpha_{ij}), \\ \mathbf{B}_{ij} &= -q'(\alpha_{ij})\cdot(\mathbf{I} - s_i s_i^*)s_j/\sin(\alpha_{ij}), \end{aligned}$$

where $\alpha_{ij} = \alpha(s_i,s_j)$ is the angle formed by s_i and s_j.

Proof: We first attempt to find the corresponding kernels of A and B:

$$\begin{aligned} (\mathrm{A}y)(s) &= \mathrm{S}\,(\mathrm{Q}y)\,(s) \\ &= \mathrm{S}\left[\int_{u\in\mathbb{S}^2} q(\alpha(s,u))\,y(u)\,d\mathcal{S}\right] \\ &= \int_{u\in\mathbb{S}^2} \left[s\times\nabla_s q(\alpha(s,u))\right]\,y(u)\,d\mathcal{S} \\ &= \int_{u\in\mathbb{S}^2} q'(\alpha(s,u))\cdot\left[s\times\nabla_s\alpha(s,u)\right]\,y(u)\,d\mathcal{S}. \end{aligned}$$

Use the definition of $\alpha = \cos^{-1}(s \cdot u)$ and the chain rule to obtain

$$\nabla_s \alpha(s,u) = -\frac{(\mathbf{I} - ss^*)u}{\sqrt{1-(s\cdot u)^2}} = -(\mathbf{I} - ss^*)u / \sin(\alpha(s,u)),$$

where the superscript $*$ denotes vector transpose, and $\mathbf{I}$ is the 3×3 identity matrix. We also have

$$s \times \nabla_s \alpha = -\left(s \times u\right) / \sin(\alpha(s,u)).$$

Therefore,

$$(\mathrm{A}y)(s) = -\int_{u \in \mathbb{S}^2} q'(\alpha(s,u)) \cdot (s \times u) \,/\, \sin(\alpha(s,u))\, y(u)\, d\mathcal{S}.$$

Compare with (18) to obtain the kernel of A:

$$a(s,u) = -q'(\alpha(s,u)) \cdot (s \times u) \,/\, \sin(\alpha(s,u)).$$

Similarly, the kernel $b(s,u)$ of B is

$$b(s,u) = -q'(\alpha(s,u)) \cdot (\mathbf{I} - ss^*)u / \sin(\alpha(s,u)).$$

The proposition can then be proved by evaluating the kernel functions at discrete directions $s_i \in \mathbb{S}^2$ according to (21). ■

REFERENCES

[1] W. Reichardt and T. Poggio, "Visual control of orientation behaviour in the fly. Part I. A quantitative analysis.," *Quarterly Reviews of Biophysics*, vol. 9, no. 3, p. 311, 1976.

[2] C. David, "Visual control of the partition of flight force between lift and thrust in free-flying *Drosophila*," *Nature*, vol. 313, pp. 48 – 50, 1985.

[3] C. Koch, *Biophysics of Computation: Information Processing in Single Neurons*. Oxford University Press, October 2004.

[4] V. Kallem, M. Dewan, J. Swensen, G. Hager, and N. Cowan, "Kernel-based visual servoing," in *IEEE/RSJ Int. Conf. on Intelligent Robots and System*, 2007.

[5] W. Stürzl and J. Zeil, "Depth, contrast and view-based homing in outdoor scenes," *Biological Cybernetics*, vol. 96, no. 5, pp. 519–531, 2007.

[6] C. Collewet, E. Marchand, and F. Chaumette, "Visual servoing set free from image processing," in *IEEE Conf. on Robotics and Automation*, 2008.

[7] A. Dame and E. Marchand, "Entropy-based visual servoing," in *IEEE Conf. on Robotics and Automation*, 2009.

[8] J. Zeil, M. I. Hofmann, and J. S. Chahl, "Catchment areas of panoramic snapshots in outdoor scenes," *J. Opt. Soc. Am. A*, vol. 20, no. 3, 2003.

[9] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *Trans. on Robotics and Automation*, vol. 12, no. 5, 1996.

[10] F. Chaumette and S. Hutchinson, "Visual servo control. I. Basic approaches," *IEEE Robotics & Automation Mag.*, vol. 13, no. 4, 2006.

[11] R. Vassallo, H. Schneebeli, and J. Santos-Victor, "Visual servoing and appearance for navigation," *Robotics and Autonomous Systems*, vol. 31, no. 1, pp. 87–98, 2000.

[12] D. Fontanelli, A. Danesi, F. A. W. Belo, P. Salaris, and A. Bicchi, "Visual servoing in the large," *Int. J. of Robotics Research*, vol. 28, no. 6, pp. 802–814, 2009.

[13] É. Marchand and F. Chaumette, "Feature tracking for visual servoing purposes," *Robotics and Autonomous Systems*, vol. 52, no. 1, 2005.

[14] A. Censi, S. Han, S. B. Fuller, and R. M. Murray, "A bio-plausible design for visual attitude stabilization," in *IEEE Conf. on Decision and Control*, 2009.

[15] E. Malis, Y. Mezouar, and P. Rives, "Robustness of image-based visual servoing with a calibrated camera in the presence of uncertainties in the three-dimensional structure," *IEEE Trans. of Robotics*, vol. 26, pp. 112 –120, 2010.

[16] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994. Also available at: http://www.cds.caltech.edu/~murray/mlswiki.

[17] F. Dellaert, S. Seitz, C. Thorpe, and S. Thrun, "Structure from motion without correspondence," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2000.

[18] A. J. Yezzi and S. Soatto, "Structure from motion for scenes without features," in *IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 1, pp. 525–532, June 2003.

[19] A. De Luca, G. Oriolo, and P. Robuffo Giordano, "Feature Depth Observation for Image-based Visual Servoing: Theory and Experiments," *Int. J. of Robotics Research*, vol. 27, no. 10, pp. 1093–1116, 2008.

[20] J. Lindemann, H. Weiss, R. Möller, and M. Egelhaaf, "Saccadic flight strategy facilitates collision avoidance: closed-loop performance of a cyberfly," *Biological Cybernetics*, vol. 98, no. 3, pp. 213–227, 2008.

[21] R. S. A. Brinkworth and D. C. O'Carroll, "Robust models for optic flow coding in natural scenes inspired by insect biology," *PLoS Comput. Biol.*, vol. 5, p. e1000555, 2009.

[22] W. B. Dickson, A. D. Straw, and M. H. Dickinson, "Integrative model of *Drosophila* flight," *AIAA Journal*, vol. 46, pp. 2150–2164, 2008.